

**INFORMATION TECHNOLOGY (IT)
SUBJECT ASSESSMENT GUIDELINES (SAGs)
(Updated January 2024)
Implementation Grade 12, 2024**

CONTENTS:

1. **Means of Assessment**
2. **Requirements**
3. **Interpretation of Requirements**

Appendix A:	Data Validation Task Mark Sheet
Appendix A.1:	Data Validation Task Component Analysis
Appendix B:	Performance Assessment Task (PAT) Mark Sheet
Appendix C	National/Regional Moderation Checklist
Appendix D:	Teacher's Record of Marks
Appendix E:	Candidate's Summary of SBA and PAT Assessment
Appendix F:	Referencing Guide
Appendix G:	Content to be covered
Appendix H:	Letter from the High School Head and IT Teachers/s
Appendix I:	Application for Alternate Language for PAT

OVERVIEW OF INFORMATION TECHNOLOGY

Information Technology is divided into the four main topics listed below.

1. **Systems Technologies**
2. **Internet and Communication Technologies**
3. **Social Implications**
4. **Data and Information Management, Solution Development**

1. MEANS OF ASSESSMENT

Paper 1 (Practical)	3 hours	150 marks reduced to 100	
Paper 2 (Theory)	3 hours	150 marks reduced to 100	(200)
Practical Assessment Task	PAT		(100)
School Based Assessment	SBA		(100)
TOTAL			(400)

2. REQUIREMENTS

2.1 INFORMATION TECHNOLOGY PAPER II (THEORY)

The theory examination will cover questions on the following sections:

1. Systems Technologies
2. Internet and Communication Technologies
3. Social Implications
4. Data and Information Management and Solution Development

Weighting of Sections

1.	Systems Technologies	40 marks
2.	Internet and Communication Technologies	45 marks
3.	Social Implications	15 marks
4.	Data and Information Management and Solution Development	50 marks
TOTAL		150 marks reduced to 100 marks

2.2 INFORMATION TECHNOLOGY PAPER I (PRACTICAL)

Solution Development, Data and Information Management

The Practical Examination will cover questions on databases using Structure Query Language (SQL), algorithms and Object-Oriented Programming (OOP). The candidate should use a text-based interface instead of a Graphical User Interface (GUI) in the practical exam. Candidates are not required to programmatically connect to the database in the practical exam.

Candidates can code their practical exam in either Java or Delphi. In 2020, C# will be included as an acceptable programming language. Candidates may use MS Access, Java DB or MySQL as a database. Please refer to IEB circulars for any changes with regards to languages and database applications.

Weighting of Sections

4.	Data and Information Management and Solution Development	
TOTAL		150 marks reduced to 100 marks

2.3 PRACTICAL ASSESSMENT TASK (PAT)

The PAT is to be completed in Grade 12, but could be started in Grade 11, and should be formatively assessed so that the candidates have the opportunity of submitting their best work.

Weighting of Sections

4	Data and Information Management and Solution Development	100 marks
TOTAL		100 marks

2.4 SCHOOL BASED ASSESSMENT (SBA)

These Subject Assessment Guidelines must be read in conjunction with the IEB Manual for the Moderation of School Based Assessment (SBA) updated 2015.

All schools must make available the SBA evidence of all candidates should it be required by the IEB or Umalusi.

The SBA assessment comprises 25% of the total assessment for the National Senior Certificate. The requirements for the school-based assessment component are outlined in the following table:

SBA Requirements

	Descriptions	Mark
Test 1	1 Practical Test	17,5
Test 2	1 Theory Test	17,5
*Task/Test	Alternative Assessment OR Test (Theory OR Practical OR Integrated)	15
Exam P1	Grade 12 Preliminary Examination Paper 1	25
Exam P2	Grade 12 Preliminary Examination Paper 2	25
Total	SCHOOL BASED ASSESSMENT (SBA)	100

*An alternative type of assessment task (Data Validation Task) for IT is given on page 20/10 of the SAGs. This type of assessment is optional to a Test.

2.6 TAXONOMIES FOR ASSESSMENT

2.6.1 Taxonomy for Information Technology Theory Assessment

The question hints below act as a guideline to indicate the complexity of the question. Words like "list", "identify" etc., imply a level 1 question however, if the content of the question is more complex, these words could imply a question that is of a more difficult level.

Level	Weight	Description	Detail
1	30%	Knowledge, Comprehension	Factual recall of content and demonstration of understanding of content listed in Appendix G; providing facts based on understanding of a simple concept as opposed to simple recall <u>Possible question types:</u> Definitions; matching columns; simple multiple choice; one-word answers to identify a term <u>Possible question hints:</u> list; identify; show; name; state; define; what is; suggest; explain
2	40%	Application, Analysis	Ability to understand the meaning of presented information; make use of given material and your own knowledge to produce an answer; use a learned concept in a new situation; comparing two situations or sets of given facts; providing facts based on understanding of a complex concept as opposed to simple recall <u>Possible question types:</u> Give advantages/disadvantages; provide an example related to a scenario; compare two solutions and recommend; simple comparison; justified suggestions related to a scenario; provide information based on user requirements; complex multiple choice <u>Possible question hints:</u> describe; contrast/compare; distinguish; discuss; illustrate; show; classify
3	30%	Abstraction (refer to Computational Thinking Appendix G – (10.4.1, 11.4.1 and 12.4.1)), Synthesis, Problem Solving, Evaluation	Ability to work through a complex scenario and understand the parts; distinguish between factual information and what is inferred; draw conclusions based on a number of factors; make suggestions to existing structures; make a judgement; give a substantiated suggestion/opinion; identify a pattern or trend from given information, algorithm, class diagrams <u>Possible question types:</u> Comparison based on multiple factors; suggest performance improvements based on scenario; show understanding of shortcomings; make a judgement based on factual content; give and justify an opinion; analyse and suggest improvements; design from given code (or reverse); code an algorithm as a solution to an unseen problem; redesign class diagrams using inheritance; improve the efficiency of a given algorithm <u>Possible question hints:</u> modify; design; assess; recommend; explain; support; compare; arrange; combine; create; rank; conclude

2.6.2 Taxonomy for Information Technology Practical Assessment

Concepts	Level 1 (30%) - Syntax, structure, command - Direct instruction (<i>Candidates are told exactly what to do. No deviation from given instructions. Total guidance to the solution/code provided</i>)	Level 2 (40%) - Multi-step procedures - Prescribed algorithm - Guided Instruction (<i>Some scaffolding/guidance to solution present.</i>)	Level 3 (30%) - Problem Solving - Adaptation of prescribed algorithm - Code efficiency – reuse of classes and inheritance - Execution efficiency – no unnecessary processing - Unguided use (<i>Little or no scaffolding provided. Development of solution given desired outcome with minimal or no instructions on how to get to the solution.</i>)
Variables	- Variable declaration and assignments of primitive and built-in types – including string - Use of constants - Array declaration of primitive types under direct instruction	- Declaration and assignments of objects under guided instruction - Declaration and assignments of an array of objects under guided instruction	- Unguided use - Code efficiency - Execution efficiency - Declaration of inherited objects under guided instruction
Operations and usage	- Operators within calculations - Use of constants and literals - Simple string function under direct instruction	Use of any combination of two control structures/methods such as a nested loop, loop with an if, array and a loop, a method with a loop	Use of any combination of three or more control structures/methods
Existing methods	- Importing and usage of additional functionality such as Maths functions (round, sqrt, etc.) - Type casting and conversions	Application or adaption of prescribed algorithms to situations (see Appendix G) requiring a combination of two or three control structures/methods such as a nested loop and an if (e.g. sort), loop with an if (finding the highest)	Unseen/unprepared problems requiring candidates to adapt prescribed algorithms significantly, or to create a unique solution to a problem.
Control Structures	- Simple selection and iteration (non-nested) - Logical operations (AND, OR, NOT) - Simple non-nested control structures (if, switch, for, do ... while, while)		Problems focusing on code efficiency and execution efficiency (e.g. "marks will be awarded for efficient solutions").
I/O	Simple I/O such as input from keyboard or output to the screen	File reading/writing within loops (e.g. splitting and converting)	- Unguided use - Code efficiency - Execution efficiency
Own method and class declaration	Declaration of void methods/procedures with no parameters	Declaration of methods with primitive or string parameters	Declaration of methods involving any

<i>(inner code classified separately)</i>	• Declaration of class under direct instruction • Declaration of standard class methods (constructors, accessor, mutator, toString) under direct instruction	• Guided declaration of static class variables and constants • Guided declaration of a class • Guided declaration of standard class methods (constructors, accessor, mutator, toString)	combination of arrays and objects • Unguided OOP design from problem definition • Guided OOP design involving a combination of arrays and objects.
SQL	• Single table, query with specified fields, and • either a single restriction (no logical operators) or sorting	• Simple queries on joined tables (using WHERE) • Use of aggregate functions • Use of functions in SQL • Statements using two joined tables with no conditions, limitations or sorting • WHERE with more than two logical operators	• Queries involving any combination of SQL in level 2 including joined tables, embedded queries and grouping • NOT IN • INNER JOIN, LEFT JOIN and RIGHT JOIN • GROUP BY, HAVING • SQL INSERT with SELECT

3. INTERPRETATION OF REQUIREMENTS

3.1 DETAILS OF PAPER II – THEORY EXAMINATION

The details for the content to be covered can be found in Appendix G of this document.

3.2 DETAILS OF PAPER I – PRACTICAL EXAMINATION

The practical exam will examine SQL statements using a database, algorithms, data structures and Object-oriented Programming concepts. The details for the content to be covered can be found in Appendix G of this document.

3.2.1 Procedure for Practical Examination

The practical examination will require each candidate to be supplied with data files. These files will be sent to schools prior to the examination. Teachers will need to check and load these files for each candidate before the exam according to the instructions that accompany the data files. Please refer to any relevant circulars regarding the delivery of data files.

3.2.2 Submission of the Practical Examinations for Marking

Please refer to any relevant circulars regarding the submission of the candidates' scripts.

3.2.3 Marking of the Grade 12 Information Technology Practical Examination

The practical examination will be marked by a marking panel in a similar manner to the theory paper.

3.3 DETAILED REQUIREMENTS FOR SCHOOL BASED ASSESSMENT (SBA) AND THE PERFORMANCE ASSESSMENT TASK (PAT)

3.3.1 Regional Moderation

Schools must submit a rank order list of the PAT marks to the Regional Moderator; the Regional Moderator will select 10% or a minimum of 5 PATs from each school. These learner PATs, together with the teacher file, must be sent to the Regional Moderator.

3.3.2 National Moderation

Schools that do not have a Regional Moderator, or at the time of regional moderation have not completed the PAT, must submit all the PATs for National Moderation.

- The submission may be made on a CD or Flash Drive, with the exception of three of the PATs, i.e. the highest mark learner PAT, the lowest mark learner PAT and any one other have to be submitted in hard copy. The CD or Flash Drive must be placed in a sealed envelope and clearly marked PATs with the school's name and centre number.

Schools that are found at the level of regional moderation to have PATs that are **compliant** with all of the requirements stipulated in the Subject Assessment Guidelines will not need to submit any PATs for National Moderation.

Schools that are found at the level of regional moderation to have PATs that are **non-compliant** with all of the requirements stipulated in the Subject Assessment Guidelines will have to submit all the PATs for National Moderation. Schools will be notified accordingly.

Please refer to any relevant circulars regarding moderation of SBA and PATs.

3.3.3 Additional information

All official queries must be sent to the IEB Assessment Specialist.

3.3.4 Mark Sheets

At the end of this document there are example mark sheets for the Data Validation Task (Appendix A) and the Performance Assessment Task (PAT) (Appendix B).

3.3.5 Formative Assessment

All informal tasks in the SBA and the PAT must be formatively assessed. When teachers review a task, they should listen to the candidate and give advice. They should be careful not to give the candidate the solution. They should suggest alternate resources and query explanations. This formative assessment is designed to help the candidate learn. It also creates the opportunity for the teacher to monitor progress, give additional input and helps guard against plagiarism. A task that has been poorly managed by the teacher can lead to substandard work by the candidate.

The PAT must follow a planned development process such as a System Development Life Cycle (SDLC) to provide candidates with a structured manner to approach large projects along with the related documentation as evidence of the candidate's process in developing the product. Teachers must discuss all planning, designs and algorithms with the candidate prior to coding and implementation. The complexity of the PAT project and choice of a suitable data structure is vital for candidates to develop a successful project.

The PAT should be completed over a number of months and must be closely monitored to avoid plagiarism.

3.3.6 Task Descriptions

The tasks should be detailed and follow the principles of assessment. The tasks should be descriptive, allow for formative assessment, provide details of deadlines, and explain how the task is to be structured. The task must give the candidate all the information required to help them produce their task. The task and the rubric must be moderated at cluster level using the principles of assessment.

3.3.7 Marking and Moderation

The contents of the SBA and PAT will be marked internally and moderated according to the IEB's moderation process. Refer to sections 3.3.1 and 3.3.2 of this document for further details in this regard.

3.4 SCHOOL BASED ASSESSMENT (SBA)

The requirements for SBA are given in 2.4.

Suggestions for tests/tasks are given below: (topics are enclosed in brackets):

Test / Task	Topic	Theory / Practical /Alternative
Theory Test	1 or 2 or 3	Theory
Normalisation Test	4	Theory
OOP Test (Written)	4	Theory
OOP Test (Programming)	4	Practical
SQL Test	4	Practical
Data Validation Task	4	Alternative Task
Preliminary Theory Examination	1, 2, 3, 4	Theory
Preliminary Practical Examination	4	Practical

The Theory Test, SQL Test, Normalisation Test and OOP Tests must be set to cover the Topics and Sub-Topics described in Appendix G. These tests must be moderated by either the cluster or another IEB Information Technology Teacher. The alternative assessment may be any one of the following:

- a theory test,
- a practical test,
- a combination of both theory and practical, or
- a task.

The preliminary examination must be set with the same weightings as those described for the November practical and theory examinations. Refer to sections 2.1 and 2.2 of this document.

All tests and examinations must be set to different cognitive levels described in the taxonomies in 2.6.

3.4.1 Cluster Set Tests

Cluster set tests must be set and moderated according to the relevant taxonomies (section 2.6) and the marking of the tests needs to be standardised. Cluster sets tests must be written on the same date to ensure

valid and reliable assessment. Should it not be possible to write cluster tests on the same day, teachers should take measures to ensure the validity of the tests.

3.4.2 Theory Test

This written test should be about 40–50 minutes in length and should contain a range of questions covering content from topics 1 OR 2 OR 3.

3.4.3 Normalisation Test

This paper must be written as a theory paper and must be approximately 40–50 minutes in length. The test must assess the normalising a single large table into more than one table up to 3NF (Third Normal Form). Include definitions of the types of normal forms, derived data, keys, anomalies, repeating groups and other normalisation terms. Candidates should be able to apply these definitions to the given table(s).

3.4.4 OOP Test (Written)

This test is to be written as a theory test and should be about 40–50 minutes in length. The content may vary, but it is recommended that a selection of the following are tested:

- A Object-oriented programming (OOP):**
 - Advantage of OOP over procedural coding.
 - Concepts of objects and classes.
 - Standard class methods (constructor, get, set, toString) and reasons for their existences
 - Data protection (private attributes with get/set method)
 - Data validation at a class level
 - Inheritance and its advantages and disadvantages.
 - Encapsulation, polymorphism (overloading and overriding) and other OOP concepts.
- B Data structure design from scenario**
 - Designing of class based on given scenario – when to create a class and when not to.
 - Use of class diagrams including inheritance
 - Use of user-defined objects as fields.
 - Understanding of sub/super class, instance field, class field, static/non-static methods.
- C Understanding of code**
 - Constructor, accessor, mutator and toString.
 - The function of inherited code and which fields/methods are inherited.
 - The function of super.
 - Compare parallel arrays vs an array of objects.

3.4.5 OOP Test (Programming)

This test is to be written as a practical test and should be about 40-50 minutes in length. The content may vary, but it is recommended that a selection of the following are tested:

- A Completing a partially coded solution**
 - Inheriting from a given class
 - Coding a class where field(s) are based on a given class (composition)
 - Coding an array of a given class
 - Coding a user interface
- B Coding a solution given a JSON file**
 - Coding a data structure to suit the JSON file
 - Reading from the JSON file and storing in the data structure
 - Manipulating the data in the data structure (adding, editing, deleting, sorting)
 - Writing to a JSON file

3.4.6 SQL Test

This test is to be written as a practical test and should be about 40-45 minutes in length and must include some advanced SQL statements using more than one table, JOINS and aggregate functions.

3.4.7 Data Validation Task (Alternate Task)

The purpose of this task is to provide a practical task to validate input using Graphical User Interface (GUI) components (radio buttons, combo boxes, check boxes etc.), exception handling and programming code with detailed error messages. A description of the task is included below:

- A Create a GUI**
Create a GUI to accept as input the following four data types
 - Numeric (e.g. What is your height or how many siblings do you have)
 - Boolean (e.g. are you male or female)
 - String (e.g. What is your first name, an Identity number)
 - Date/Calendar (choose an appropriate date using the date picker)
 using the most appropriate component for each data type.
- B Select appropriate GUI components**
Select the most appropriate GUI component to eliminate incorrect input and provide the first level of data validation.
- C Data validation using program code**
Provide a further level of validation using programming code (and possibly exception handling) to perform any FOUR of the following validation checks on one or more field. Note that some fields may not require further validation checks and can be sufficiently validated using the appropriate GUI component. Note that the Date type can be validated using methods from existing classes provided by the programming language.

Validation Type	How it Works	Example Usage
Check Digit	The last one or two digits in a code are used to check the other digits are correct	Bar codes, ISBN numbers, credit card numbers
Format Check	Check the data is in the right format	A number plate being ABC 999 GP
Length Check	Checks that the data is not too long or too short	A password is ten characters long
Lookup Table	Looks up the acceptable values in a table	There are only seven possible days in the week
Presence Check	Checks that data has been entered into a field	An ID number cannot be blank
Range Check	Checks that a value falls within the specified range	The number of hours worked can be between 0 and 50
Type Check	The data entered is the correct type	The value for a height is a real number.
Logic Check	The value entered is logically correct	A person in Grade 12 cannot have an age less than 16 or greater than 19.

D Example using one field

Note that the four validation rules can apply to more than one field:

STRING: Identity Number

- Format Check – All digits present, the first 6 are a valid date
- Length Check – 13 digits
- Presence Check – available or not
- Check Digit – last digit is correct

E Possible GUI design

DATA VALIDATION TASK

1. Enter a data type Grade (R, 1 - 12)

2. Enter a String Identity Number

3. Enter a Boolean ☐ Male ☐ Female Gender

4. Choose A Date Date of birth

F Error messages

The program must include a detailed error message for each and every error. Incorrect values must be indicated by highlighting the field's component and describing the error in an adjacent label. If all fields are acceptable, then a **label** must display a message indicating all input is correct.

G Data validation document

Submit a Data Validation document as evidence of your task. Your document must consist of the following:

- (i) Title page including the candidate's name, examination number, centre details and title of the task, table of contents and page numbers
- (ii) An image of the GUI before data is entered.
- (iii) A completed table included in Appendix A.1 that describes each field to be input, the type of GUI component chosen for this field, the reason for the choice of GUI component and type of data validation that is performed using programming code.
- (iv) A test plan for each validation rule using standard, extreme and abnormal data.
- (v) Evidence of testing using before and after screen shots.
- (vi) A copy of all programming code with comments explaining each validation rule.

3.5 PERFORMANCE ASSESSMENT TASK (PAT)

The purpose is to give the candidate a meaningful experience of a larger project to combine the skills taught in programming.

3.5.1 Scope

The project must include permanent storage of data, a user interface (frontend) and classes to store and manipulate the data (backend). This project should be more complex and include more features than those of a practical examination paper. It should not be something that the candidates can produce in as little as a week. Inappropriate examples might include a calculator, the Hangman game or a currency converter.

3.5.2 Topic and Content

The project topic may be a game, a business management system or a solution to a real-world problem such as the school tuck shop including ordering, storage of appropriate (see POPI Act) student data and calculations of profits.

The project can be implemented on one or more desktop computers, as a mobile application or on a separate hardware device such as a Raspberry Pi or Arduino, or any combination thereof. Teachers are to ensure that candidates have written a substantial amount of their solution and have not relied too heavily on utilities and functions provided by the solution development tool such as Unity in game development.

In order to reduce the risk of plagiarising an existing project of which there are many available on the Internet, it is advisable to design and code a complete new game, or design and code a solution for a new situation. It is vital that any external code used by the candidate does not comprise more than 20% and is suitably referenced.

Candidates are expected to research existing solutions to their project to determine whether there is an existing solution. Any project that has a complete solution in existence should be avoided.

It is recommended that candidates consider solutions to national or global problems described in the National Development Plan 2030 (NDP 2030). These problems described in the NDP are related to the 17 global goals listed in the Sustainable Development Goals (SDGs) set by the United Nations.

3.5.3 Suitable Programming Languages

The program must be written in one of the programming languages approved by the IEB. (See the section 2.2 of this document and relevant IEB circulars with regard to the languages currently approved by the IEB for use with practical work and the PAT.) Should a candidate wish to code their project in an alternate language, permission needs to be obtained from the Portfolio Moderator in writing by 28th February of the candidate's matric year. The teacher needs to inform the IT Subject Specialist and Portfolio Moderator using Appendix I. The teacher needs to acknowledge that the candidate's

portfolio (including their PAT) is sent to NATIONAL MODERATION to verify that the PAT is of a suitable standard. The teacher must ensure that the project has been marked with the IEB mark sheet in Appendix B.

The project needs to meet the following criteria:

A Problem Specification and Motivation

The project should aim to solve real-world problem as far as possible. It should not be contrived or generic. For examples, games should not just provide entertainment; they should deliver functional purpose to a group of end-users – not just one or two people or just the author. There should be some form of evaluation with respect to how far the solution provided solves the proposed problem with some member of the group of end-users. This requires that at least one member of the target group of users is available for this purpose.

Existing similar solutions should be investigated. For example, the candidate may create a card game based on the card game Black-Jack. The candidate should briefly describe existing solutions that exist specifying the type of solution (mobile application or PC based), programming language, multiplayer options etc. The motivation should describe why the candidate has chosen to reproduce such a game and how their game will be different. Teachers should be aware that if the candidate chooses to create a game that is significantly similar to an existing the game, they run the risk of producing a substandard project or plagiarising the existing project.

B User Interface

The user interface can be either a GUI or a text-based user interface. It must be easy to use and task appropriate. If the program does not use a GUI there has to be a good, task appropriate reason for this (which is adequately explained in the design document). A hardware-based project such as Raspberry PI or Arduino must still provide a user interface whereby the user can interact with the program. Teachers must guide the student, ensuring that the project has adequate interaction with the user or they will risk receiving a lower mark for this section.

C Other Interfaces

Other input and output via sensors, network connections, cloud sources etc. need to be specified.

D Sequencing/Data flow/Program operation

The candidate must ensure that the sequence of steps required to use the program and complete a task are clear, easy to follow and logical.

E Storage/Data Persistence

Data must be stored and retrieved from session to session which can be in the form of conventional files (such as text files or JSON files) OR a database OR a combination thereof. The storage must be appropriate to the program for example, a game should have the ability to save, load and determine high scores.

F Algorithm Complexity

The project must showcase the candidate's programming ability that is of Grade 12 level or beyond in terms of problem solving (refer to levels 2 and 3 in section 2.6.2). This complexity level is accessed in program operation, data processing and data storage. Candidates must ensure that sufficient level of complexity is distributed among these. For example, an application or game that is light on data storage must contain more complex processing algorithms in order to satisfy this criterion.

G Separation of Interface and Engine

The 'working code' **must not be embedded in the interface** (i.e. it must be in separate classes/units). Communication between the interface and the working code must be in the form of parameters and typed methods/functions. A limited amount of code in the interface is acceptable only if suitably justified in the planning.

H Good Data Internal Structures

The data in the project must be internally represented using classes/arrays – or any combination of these. Data structures must be logical and task appropriate. Classes are essential.

To be fair to the candidate and to ensure a good product at the conclusion of the project it should be completed and assessed in 5 phases.

Phase		Description	Marks
What must be submitted by the candidate:	1 – Specification Document	Specification of the problem, user interface, data storage and hardware requirements. List (and describe) the functions that your program needs to achieve in order to be a 'success'.	17
	2 – Design Document	Design the user interface, sequencing (data flow), class and persistent storage of the program in detail.	30
	3 – Coding	Write the program following good programming techniques.	38
	4.1 – Technical Document	Document the project by printing the code and explaining critical algorithms.	8
	4.2 – Testing Document	Document what is to be tested, the test data used and the results of the testing.	7

In order to give the candidates a starting point, **teachers may provide templates** that show the type of content required in the **Project Specification, Design, Technical** and **Testing** documents. Candidates are to be encouraged to use these templates to reduce workloads, standardise the PAT and simplify marking/moderation.

Refer to the Mark Sheet – Appendix B

APPENDIX A



**NATIONAL SENIOR CERTIFICATE EXAMINATION
INFORMATION TECHNOLOGY
DATA VALIDATION TASK MARK SHEET**

The purpose of this task is for candidates to demonstrate their ability to create a GUI to input four different data types. Candidates are to validate the data types using the appropriate GUI components and perform any FOUR of the following data validation checks using exception handling and/or programming code. Each of the four checks must be tested using standard, extreme and abnormal data. Standard data has values within the allowable range, extreme data has values on the limits of the range and abnormal has invalid values. For example, if the allowable range is from 1 to 10, standard data would be 4, extreme data 10 and abnormal data -2, 'p' or F7.

Criteria	Description	Possible Mark	Actual Mark
Interface Design Requirements (GUI) Screen is clearly laid out, with clear font, colour, size of font, and a button to perform validation (3) Fields aligned and clearly labelled (2). Screen has a title and centred on the screen (1). Each input component has an associate label (2)	<u>Use the four categories to allocate marks.</u> -1 for each error.	8	
Variety of Types Used Four Data Types present on GUI – String, Numeric, Boolean and Date/Time.	Deductions to a maximum of 4 (-1 for each type of missing field)	4	
Choice of appropriate components for each data type Numeric, String, Boolean, Date/Time	Deduction to a maximum of 4 (-1 for each missing component)	4	
Valid reason for choice of each component Numeric, String, Boolean, Date/Time	Deduction to a maximum of 4 (-1 for each missing or incorrect reason)	4	
Naming Conventions Labels, buttons, fields all correctly named according to conventions.	Deductions to a maximum of 2 (-1 for each incorrectly named field)	2	
Programming code with possible exception handling for each validation rule Correct, working code for each of the four validation rules. (4×2) Rules achieve what is described in the reason for the rule (4×1).	3 marks for each rule. -1 to a max of 3 for each error in each rule.	12	
Descriptive error message for each rule Error message is provided with a detailed explanation. (4×1). Each error message is placed in a label next to the incorrect component. (2) If all input is valid, a confirming message is displayed (2) Do not penalise twice for an error above.	Deduction to a maximum of 2 per incorrect error message. Deduct 1 mark for a vague message per rule.	8	
Testing for each rule Choice of standard, extreme and abnormal data to test each rule (1×4) with evidence of testing (1×4)	Deduct 1 mark for incorrect data per rule Deduct 1 mark for lack of evidence per rule	8	
TOTAL		50	

APPENDIX A.1



**NATIONAL SENIOR CERTIFICATE EXAMINATION
INFORMATION TECHNOLOGY
DATA VALIDATION TASK COMPONENT ANALYSIS**

The first row includes an example of an age field. Please do NOT use this example, as this value can be calculated from a date.

Name of Field and Type	Input Component	Reason for Input Component	Data Validation – Name of Rule (e.g. Format, length, presence, check digit, range, type, logic)	Reason for Data Validation
Example: age: integer (Do NOT use this example)	Text field	User can enter any age	Range Check	Age must lie between 0 and 125

APPENDIX B



**NATIONAL SENIOR CERTIFICATE EXAMINATION
INFORMATION TECHNOLOGY
PERFORMANCE ASSESSMENT TASK – MARK SHEET**

1. SPECIFICATIONS DOCUMENT (See SAGs Appendix G – 12.4.16)	MAX	ACTUAL	COMMENTS
1.1 Problem Summary A brief description of the project including the purpose of the project, summary of functions and description of target user group(s)	3		
[3] Purpose, summary of functions and target user groups, well-described.			
[2] Aspects mostly well-described, with at least one not completely specified/described.			
[1] Only one aspect well described or part of each not fully described.			
[0] Aspects not described or completely inadequate.			
1.2 Research and Motivation Research is done to discuss similar projects. Research is correctly cited and referenced. A motivation is included for the project indicating how this project differs from existing projects.	3		
[3] Similar projects are described, cited and referenced. A motivation is supplied to include an explanation of how the proposed project will differ from existing projects.			
[1-2] Incomplete research, citing, referencing or insufficient motivation. Deduct one mark for each incomplete aspect.			
[0] No research or motivation provided.			
1.3 Program Functions (See SAGs Taxonomy 2.6.2)	3		
[3] Function list for scenario is detailed with sufficient complexity for the proposed program.			
[2] The function list is a substantial list of appropriate outcomes but is insufficient in complexity or the functions list is not complete.			
[1] The function list results in a simplistic program.			
[0] No functions listed.			
1.4 User Interface (See SAGs Appendix G – 10.2.5, 10.4.12, 11.4.12) Specify the user interface and if relevant, the input/output from external hardware.	2		
[2] User Interface completely specified for given scenario and stipulated program functions.			
[1] User Interface for given scenario incomplete. Maximum of two items inadequately specified for given scenario and stipulated program functions.			
[0] User Interface not specified or incorrectly specified.			
1.5 Help Features (See SAGs Appendix G – 12.4.15)	2		
[2] A variety of help features available in the program, to assist the user. Features listed and well described.			
[1] Some help features available in program, listed and partially described.			
[0] No help features in program.			

1.6 Permanent Data required		2		
[2]	All data for given scenario and program functions, have been correctly described and grouped appropriately.			
[1]	Some data for given scenario and program functions are described with a few errors, or data is not appropriately grouped (i.e. related fields should be listed together).			
[0]	No data provided for given scenario and program functions.			
1.7 Hardware and Software Requirements (including additional hardware)		2		
[2]	Complete list of appropriate hardware (capacity/speed/size) and software specifications (with versions), for both user and developer.			
[1]	Hardware and/or software specifications not complete. Missing details for hardware and/or software.			
[0]	Hardware and software specifications not discussed.			
TOTAL		17		

2. DESIGN DOCUMENT (see SAGs Appendix G – 12.4.16)	MAX	ACTUAL	COMMENTS
2.1 User Interface Design (See SAGs Appendix G – 10.2.5, 10.4.12, 11.4.12) ALL screens for the user must be completely specified in this section. All required data on screens must be in relevant GUI components. All action elements on screens (with input devices) must be listed and clearly described. Screen capture from a design tool (including IDE), sketches and mock-ups are acceptable.	6		
[5-6] Sufficient level of user interface design present with consideration given to good design principles for an effective user interface. Correct components used for required data, on each screen. Correct action elements, with input devices, on each screen, listed and described in detail.			
[3-4] Sufficient level of user interface design present with some consideration given to good design principles for an effective user interface. Some incorrect components used for required data, on screens. Some incorrect action elements or incorrect input devices or action elements not described in detail, on screens.			
[0-2] No user interface design present or no consideration has been given to good design principles for an effective user interface. Incorrect components used for required data on screens. No action elements and input devices or not described in detail, on screens.			
2.2 Program Flow Diagram Use a flow diagram or any other form of illustration to present a global overview of how the program is used.	5		
[4-5] Flow is clear, well represented and easy to understand. No logical gaps are evident.			
[2-3] Flow is substantial but still has some logical gaps.			
[0-1] No flow and/or large logical gaps.			
2.3 Class Design and OOP Principles (see SAGs Appendix G – 11.4.3, 12.4.3, 11.4.5, 12.4.5) The candidates must provide their class design represented as a UML class diagram with class name, fields, and methods demonstrating the application of OOP principles. Only provide backend classes NOT user interface classes.	8		
<u>Class Design</u> [3] Class design is thorough – all fields and methods are present. Fields and methods clearly relate back to the Specifications Document.			
[1-2] Class design is substantial but shows obvious gaps in missing fields/methods or has minor errors.			
[0] No class design or class design is incorrect or is rudimentary with little detail. Fields are incomplete, methods are minimal/not well thought out.			
<u>OOP Principles</u> [4-5] Fields and methods are separated logically, into classes. Fields and methods are protected sensibly. Good use of OOP principles where necessary.			
[2-3] Fields and methods are separated logically, into classes. Some instances of incorrect or inappropriate use of OOP principles.			
[0-1] No attempt to separate into classes. Some attempt at a class diagram with little to no organisation.			

2.4 Secondary Storage Design (see SAGs Appendix G – 11.4.7, 12.4.7, 10.4.10, 12.4.10, 10.4.11, 11.4.11, 12.4.11) Candidate must show how data structure in primary memory described in section 2.3, will be permanently stored. Storage design should be done using tables in a database, text files, JSON files or a combination thereof. Storage can be local, remote or cloud based. For a database, screenshots of tables with record structure and field types from database software are acceptable along with sample data for each table. For text files, an explanation of the structure of the file must be explained together with sample data.	5		
[5] Storage design is well described – fields are listed with data types and description. Storage design is appropriate to purpose and matches the Specification Document. There are no missing aspects.			
[3-4] Storage design is well described but with a few missing aspects.			
[1-2] Storage design is evident, but description is superficial/vague/incomplete or with errors.			
[0-1] No storage design evident or storage design is rudimentary.			
2.5 Explanation of Secondary Storage Design (See SAGs Appendix G – 12.4.7, 12.4.10) The candidate must provide an explanation of their secondary storage design. The explanation must demonstrate a justification of the secondary storage design and an understanding of the implications of the chosen design as opposed to other storage designs.	3		
[3] Explanation shows in-depth understanding of the implications of the secondary storage design and is completely justified.			
[2] Explanation is substantial, but it is not completely justified. There are some areas of confusion or lack of understanding of the implications of the storage design.			
[0-1] No explanation of secondary storage design is provided or no evidence of understanding of the storage design.			
2.6 Explanation of how Primary Data Structures relate to Secondary Storage (See SAGs Appendix G – 10.4.3, 11.4.3, 12.4.3, 11.4.7, 12.4.7) Description of how the primary data structures described in class diagrams (assessed in section 2.3), will represent the secondary storage design (assessed in section 2.4). There should be a description for each backend class listed in section 2.3, that will translate to how data is sent to and from secondary storage.	3		
[3] A clear and detailed representation of which class relates to which secondary storage data and how the data will be represented, when the data is read from or written to secondary storage.			
[1-2] Some form of representation of which class relates to which secondary storage data and how the data will be represented, when the data is read from or written to secondary storage, however there are missing details.			
[0] No representation of which class relates to which secondary storage data.			
TOTAL	30		

3. CODING This is assessed by examining the source code. The project must be run to determine whether it achieves the functions listed in section 1.3.	MAX	ACTUAL	COMMENTS
3.1 Comments (See Appendix G – 12.4.15) Code is commented using an API and/or comments which are placed inside source code to explain code, parameters and return types. Only backend classes/units can have APIs and all other classes/units need to be commented. Comments need not be provided if descriptive names are used for methods/functions, fields and variables. [3-4] All required code is present, with comments. All methods/functions have comments describing what they do. Comments include the data they return (for typed methods/functions) and the data they receive (parameters). Steps in complex algorithms are commented. [1-2] Most of the required code is submitted and/or only some code has comments. Not all methods/functions are commented. Comments are brief and contain little relevant detail. Parameters and return types are not all commented. [0] Most of the required code is not submitted and/or there are no comments.	4		
3.2 Separation of UI from Working Code (See Appendix G – 11.4.5, 12.4.5) [4-5] Complete separation of all required code. Different classes/units are separated in the backend, from the UI and other interfaces. The backend classes/units can be 'plugged into a different UI that uses all the methods/functions appropriately. [1-3] Some separation of all required code. There are separate classes/units, but work based on data and secondary storage, is still done in the UI. Insufficient further breakdown and separation in the backend classes/units. [0] Most of the required code is not submitted and/or no separation of code from the interfaces.	5		
3.3 Inter-Code Communication Typed Methods/Functions and Parameters (See Appendix G – 10.4.6, 11.4.6, 12.4.6) Inter-code communication occurs between classes and within a class between methods. [4-5] There is effective and conceptually correct use of typed methods/functions and parameters. [1-3] Some use of typed methods/functions and parameters. Marks can be deducted as follows (-1 per every unique error type, multiple instances of the same error do not accumulate deductions): Errors include: unnecessary use of parameters, incorrect parameter types, parameters specified but not used, incorrect typed method/functions, failing to return values in typed methods/functions, failing to use the results returned by typed methods/functions, using variables/fields where the value is best returned by a typed method/function. [0] No inter code communication, no typed methods/functions or parameters.	5		
3.4 Good General Programming Techniques (See Appendix G – 10.4.6, 11.4.6, 12.4.6, 10.4.8, 11.4.8, 12.4.8) [5] Code is technically perfect. Indentation immaculate. Variable names are all descriptive and follow conventions. Programming structures are appropriate e.g. switch instead of if statements, duplication of code is eliminated using appropriate structures such as arrays or methods. [1-4] Errors in programming techniques (-1 per error type – multiple instances of the same error do not accumulate deductions). Errors include: No indentation, single level indentation, inconsistent or inaccurate indentation, variable names do not clearly indicate what the variable is used for, multiple variables used instead of arrays, multiple if statements instead of switches or case statements, repetition of code (instead of using a typed method/function/void method/procedure arrays). [0] Code does not incorporate any good general programming techniques.	5		

3.5 Querying and Manipulation of Data in Secondary Storage (See Appendix G – 11.4.7, 12.4.7, 10.4.11, 11.4.11, 12.4.11) This section refers to the implementation of the primary data structure discussed in section 2.3 and the secondary data storage discussed in section 2.4, and the representation of the secondary storage in the primary data structure as discussed in section 2.6.	6		
<u>Primary data structure implementation</u> <i>evaluate against what was specified in section 2.3</i> [2] Implemented fully as described. [1] Implemented but not fully as described. [0] No implementation of specification.			
<u>Secondary storage implementation</u> <i>evaluate against what was specified in section 2.4</i> [2] Implemented fully as described. [1] Implemented but not fully as described. [0] No implementation of specification.			
<u>Implementation of secondary storage representation</u> <i>evaluate against what was specified in section 2.6</i> [2] Implemented fully as described. [1] Implemented but not fully as described. [0] No implementation of specification.			
3.6 Defensive Programming (See Appendix G – 11.4.7, 11.4.8, 11.4.14) Data validation, exception handling, error messages for all interfaces.	4		
[4] Data is controlled and validated using code or appropriate UI components. Candidate need only code once, for every data type instance, where UI components are not used. Potential major IO and Mathematic errors protected/trapped with relevant exception handling. All error messages are descriptive and easy to understand.			
[3] Most data are controlled and validated using code or appropriate UI components. A few instances where candidate has not coded for every data type instance, where UI components are not used. Potential major IO and Mathematic errors protected/trapped with relevant exception handling. Vague or insufficient error messages.			
[1-2] Few data are controlled and validated using code or inappropriate UI components used. Many instances where candidate has not coded for every data type instance, where UI components are not used. Potential major IO and Mathematic errors not protected/trapped with relevant exception handling. No error messages.			
[0] No data validation and exception handling.			

3.7 Fulfilment of Specifications The project must be tested against the functions listed in section 1.3. This can only be assessed by running the compiled program. Cross reference with the functional testing done in section 4.2.2		5		
[5] All functions listed, work correctly.				
[4] 90% of functions listed, work correctly.				
[1-3] Basic implementation of functions listed. Missing functions or significant number of functions do not work.				
[0] Program does not execute.				
3.8 User Experience (See Appendix G – 10.2.5, 10.4.12, 11.4.12) This can only be assessed by running the compiled program.		4		
[4] Program is easy to use, navigate and understand: an excellent user experience.				
[2-3] Sufficient level of user interface present to provide a good user experience for the program. Some instances of unnecessarily complex navigating and/or aspects design that are confusing to the user.				
[1] The user is lost – does not know where to start or how to use the program, or user interface present but not at a sufficient level for the user to engage with.				
[0] Program does not execute or there is no user interface to engage with.				
TOTAL		38		

4.1 TECHNICAL DOCUMENTATION (see Appendix G – 12.4.16)	MAX	ACTUAL	COMMENTS
4.1.1 Externally Sourced Code This must be present even if the candidate only declares that no external code has been used. No more than 20% of the code may be from an external source.	1		
[1] Candidate has declared externally sourced code. This can be confirmed with an interview incorporating oral review of code and techniques.			
[0] Not present.			
4.1.2 Explanation of Critical Algorithms The core algorithms that are critical to the correct functioning of the program. There may be a few, or even only one critical algorithm. Each algorithm must be done using correct convention and must have an explanation. Programming code is NOT acceptable.	3		
[3] Algorithm(s) with correct IEB convention, and clear explanation(s) as to why algorithm(s) is critical.			
[1-2] Algorithm(s) contain errors or not done according to IEB convention and/or explanation(s) as to why algorithm(s) is critical, is not clear.			
[0] No algorithms(s) and explanation(s).			
4.1.3 Advanced Techniques A minimum of TWO techniques that are NOT part of the syllabus. The code and an explanation for each technique, must be provided.	4		
[3-4] A good explanation of at least TWO techniques not in the syllabus. Code for techniques is not superficial.			
[1-2] Candidate has listed techniques in the syllabus or techniques not clearly explained or code for techniques are superficial. Only one significant advanced technique is included.			
[0] Not present.			
TOTAL	8		

4.2 TESTING DOCUMENTATION (see Appendix G – 12.4.16)	MAX	ACTUAL	COMMENTS
4.2.1 Evaluation of the Programmed Solution An objective evaluation on how the programmed solution satisfies the functions listed in section 1.3. Suggestions must be included on how to address failures, should functions not be met. Alternate solutions or improvements must also be included.	2		
[2] Evaluation is thorough and includes suggestions for all shortfalls and/or suggestions for improvements.			
[1] Evaluation includes suggestions for some shortfalls and/or suggestions for improvements, not clearly explained.			
[0] No evaluation.			
4.2.2 Functional Testing: At least TWO sets of functional testing evident together with the tester's name, the date the testing was performed and the result of each functional test. Each test should indicate whether the project satisfies the functions listed in section 1.3. It is acceptable if some of the functions are not working as long as progression is seen through the testing process.	3		
[3] TWO sets of functional tests and all requirements tested with all details present.			
[1-2] Not all requirements were tested and/or not sufficiently described: missing details such as when, with whom and result.			
[0] No testing, not indicated or original function list was insufficient (or not present).			
4.2.3 Test Plan and Results for TWO input variables (see Appendix G – 10.4.13, 11.4.13) The TWO input variables must be clearly identified. Testing should be done using standard, extreme and abnormal data. Screenshots showing before and after of each test for each variable must be included.	2		
[2] Full test plan and results present for TWO input variables which are clearly identified together with screenshots.			
[1] Some test plan and result present but at least one element is missing.			
[0] No test plan and no result present.			
TOTAL	7		

APPENDIX C



**NATIONAL SENIOR CERTIFICATE EXAMINATION
INFORMATION TECHNOLOGY
NATIONAL/REGIONAL/MODERATION CHECKLIST**

SUBJECT: INFORMATION TECHNOLOGY		CENTRE NUMBER			
Teacher's Name		School			
Moderator's Name		School			
Teacher's portfolio available?					Yes/No
TEACHER'S PORTFOLIO – GENERAL		Programming Language Used?	Delphi	Java	C#
Cover sheet with centre's details clearly labelled?	Yes/No	Evidence of attended cluster meetings?			Yes/No
Appendix C?	Yes/No	Appendix D?			Yes/No
PAT Task sheet available?	Yes/No	PAT marking guidelines available?			Yes/No
Theory Test available?	Yes/No	Theory Test marking guidelines available?			Yes/No
Practical Test available?	Yes/No	Practical Test marking guidelines available?			Yes/No
Task/test available?	Yes/No	Task/test marking guidelines available?			Yes/No
Preliminary theory examination available with the paper analysed to cognitive levels?	Yes/No	Preliminary theory marking guidelines available?			Yes/No
Preliminary practical examination available with the paper analysed to cognitive levels?	Yes/No	Preliminary practical marking guidelines available?			Yes/No
TEACHER PORTFOLIO – TASKS AND TESTS					
The standard of the Theory test (SBA)	Comment:	too easy	easy	appropriate	too difficult
The standard of the Practical test (SBA)	Comment:	too easy	easy	appropriate	too difficult
The standard of the task/test (SBA)	Comment:	too easy	easy	appropriate	too difficult
The standard of Preliminary Practical exam (SBA)	Comment:	too easy	easy	appropriate	too difficult
The standard of Preliminary Theory exam (SBA)	Comment:	too easy	easy	appropriate	too difficult

LEARNER PORTFOLIOS – MARKING (if learner portfolios required for moderation)				
Marking according to memo?	Yes/No	Allocation of marks justified?	Yes/No	
LEARNER PORTFOLIOS – RECORDING (if learner portfolios required for moderation)				
Learner achievement recorded?	Yes/No	Appropriate aggregation?	Yes/No	
LEARNER PORTFOLIOS – PAT				
Correct use of parameters and subprograms?	Yes/No	Correct use of program structure – sequence, selection (if/case) and iteration (loops)?	Yes/No	
Project divided into Classes/Units?	Yes/No			
Correct documentation?	Yes/No	Project based on a central theme?	Yes/No	
Standard of the performance of the PAT			too easy	easy
			appropriate	too difficult
LEARNER PORTFOLIOS – SBA (if learner portfolios required for moderation)				
Prelim Theory and Practical scripts included?	Yes/No	Scripts have been accurately assessed?	Yes/No	
Theory Test scripts included?	Yes/No	Scripts have been accurately assessed?	Yes/No	
Practical Test scripts included?	Yes/No	Scripts have been accurately assessed?	Yes/No	
Task/Test is included?	Yes/No	Scripts have been accurately assessed?	Yes/No	

Additional Comments:

Describe any interesting/innovative work:

TEACHER'S NAME		DATE	
TEACHER'S SIGNATURE			
MODERATOR'S NAME		DATE	
MODERATOR'S SIGNATURE			

IEB Copyright © 2014–2025

APPENDIX E



**NATIONAL SENIOR CERTIFICATE EXAMINATION
INFORMATION TECHNOLOGY
CANDIDATE'S SUMMARY OF SBA AND PAT ASSESSMENT**

(To be filled in by the candidate and controlled by the teacher. To be included as the **1st** Page of the learner's SBA)

Centre Number:

Examination number:

Description of task	Brief Description	Possible Mark	Actual Marks (Calculated to TWO decimal places)
PAT		100	
PAT – 100 (Round to nearest integer)			
Theory Test		17,5	
Practical Test		17,5	
Task/Test		15	
Prelim Practical Exam		25	
Prelim Theory Exam		25	
SBA – 100 (Round to nearest integer)			

DECLARATION BY THE CANDIDATE:

I, _____ (print full names) declare that all the external sources used in my SBA and PAT have been properly referenced and that less than 20% of the code in my PAT has been obtained from external sources, as required by the IEB.

Signed: _____
Candidate

Date: _____

DECLARATION BY THE CANDIDATE'S TEACHER:

I _____ (print name and title of teacher) at
_____ (print name of school) declare that the work provided by this candidate has been monitored and checked for plagiarism.

Signed: _____
Teacher

Date: _____

APPENDIX F



NATIONAL SENIOR CERTIFICATE EXAMINATION INFORMATION TECHNOLOGY REFERENCING GUIDE

You will find full details of the Harvard Standard on the Sheffield University site at:
<<https://www.mendeley.com/guides/harvard-citation-guide>>

A book should be referenced as follows:

Allen, John R. Burke, Michael E. and Johnson, John F. Allen, 1983: *Thinking about Logo*. Holt Rinehart and Winston New York

1. The author's surname, followed by a comma, followed by the first names or initials. If there is more than one author then note all of the authors.
2. The date of the publication.
3. The title in italics.
4. The publisher's name and location.

A webpage should be referenced as follows:

Baldwin, Richard Java *2D Graphics, Simple Affine Transforms*, 2003. Available from:
<<http://www.developer.com/net/cplus/article.php/626051>> [Accessed 17th October 2003]

1. The author's surname, followed by a comma, followed by the first names or initials. If there is more than one author, then note all of the authors.
2. The date of the publication.
3. The title in italics.
4. The words "Available from:" followed by the URL.
5. In square brackets the word 'Accessed' followed by the date when the site was visited. (Candidates must be aware that the date of their Web site visits must be noted for the bibliography.)

A list of references

A list, in alphabetical order of author, giving details of the sources quoted in the text, as described above, for books and Web sites, must appear at the end of the task under the heading "References".

Please note the Harvard standard is not the only acceptable way of referencing but it is the one preferred by a number of universities in South Africa. **Whichever method is used, it must be used consistently.**

APPENDIX G



NATIONAL SENIOR CERTIFICATE EXAMINATION INFORMATION TECHNOLOGY CONTENT TO BE COVERED

Content Description

The following content list is divided into the four topics and shows the learning progression from Grade 10 to Grade 12.

Each topic is divided into subtopics and is allocated a unique number. The format of the number includes the Grade, topic number and subtopic number, for example, 10.1.3 is the subtopic 3 in Grade 10 of topic 1. Each subtopic consists of a skills header followed by a list of content.

Topic 1: System Technologies – Hardware and Software					
Grade 10		Grade 11		Grade 12	
10.1.1	<i>differentiates between the concepts of hardware, software and connectivity, data and information.</i> - Definition/description of hardware/software - Definition/description of ICT system - Generic model/definition of a computer – Input Processing Output Model (IPO) - Advantages and disadvantages of using computers - Explanation of and differentiation between data and information				

Topic 1: System Technologies – Hardware and Software					
10.1.2	identifies and distinguishes between computer types and associated software. System types: - Laptops, desktop, server, embedded computers, smart wear, tablets, smartphones, single board computer e.g. Raspberry PI and Arduino Classification of computing devices: - Portable/Mobility - Processing power – server, super computer, desktop, mobile Operating systems associated with: - Desktop OS - Mobile OS - Embedded OS Application software: - Stand-alone applications - Network applications Data transfer and synchronising between devices				
10.1.3	identifies the main hardware components of computing devices. Motherboard components: - CPU – ALU, CU, registers - Primary Storage (BIOS, RAM, ROM) - Secondary Storage (Mechanical hard drive (HDD), solid state drive (SSD), hybrid drive, flash drives, optical, SD cards) - Input devices (pointing devices, keyboard types, scanners, microphones, biometric devices, sensors e.g. accelerometer) - Output devices – monitors, printers, speakers, headphones - Ports: USB, HDMI	11.1.3	discusses how the various components of computers interact with one another. CPU design: - Parallel processing – hyper-threading and multi-processing - Registers (effect of number of bits), ALU, CU Concepts of: - Difference in performance of CPU and RAM-speed, latency - SRAM (processor cache and registers) - Purpose and types of caching (processor, disk, browser, proxy/web) Motherboard function and connection of components: - System Clock to synchronise events - Over clocking and clock multiplication - Internal Bus/FSB - data bus, address bus and control bus and relationship to registers in the CPU - External Buses – speed vs. throughput: PCI express, SATA, USB	12.1.3	analyses factors affecting overall performance of a computer-based system. Factors that affect performance and reliability: - Modular design: ports, cards vs onboard, buses - Co-Processors e.g. Graphic processor, Maths Processor - Techniques to improve processor speed: - Hyper-threading and multi-processing - Increasing level 1,2,3 cache, register size, effect of register size on data and address busses Changing a component's speed via clock multiplication and/or overclocking - Increasing speed and size of RAM - Improving other components/devices specific to the task: - Video card for 3D rendering - Faster HDD/SSD for video editing

Topic 1: System Technologies – Hardware and Software					
		11.1.4	compares primary and secondary storage in terms of speed, bandwidth, capacity and reliability. Categorises primary memory: - Registers, CPU cache, RAM Categorises secondary storage: - Flash memory, HDD, SSD, external hard drives, cloud storage	12.1.4	makes recommendations for a hardware solution for a given problem. - Motivate a typical computer system with respect to the hardware needed for a specific purpose computer system - Motivate a computer-based solution to a specified problem as it relates to the needs of specified user - Describe mobile technologies and constraints: - Battery life, size, computing power, power consumption (see 11.2.6)
10.1.5	identifies the functions of various types of operating system. General functions of a generic operating system: - User interface - Load and run programs - Manage resources - Interface between hardware and application programs	11.1.5	discusses processing techniques and memory, storage and IO management. Machine Cycle: - Fetch, Decode, Execute and Store Identify, analyse, compare and recommend processing techniques: - Software – Multi-tasking, Multi-threading related to: - Hardware – Hyper-threading and multi-processing UEFI vs BIOS and CMOS Interrupts - IRQs and IO Range Virtual memory: - Describe virtual memory including paging and swapping - Effect of virtual memory on processing speed		
10.1.6	distinguishes between the types of system software. Operating systems: - Utilities - Drivers - Programming tools and related applications - Source code vs executable code/bytecode - Distribution and licensing models Open source software: - Proprietary software - Freeware - Freemium software - Creative commons	11.1.6	discuss and analyse types of programming tools. Compare and recommend language translators: - High level languages - Low level languages - Compilers (one and two stage), interpreters and assemblers		

Topic 1: System Technologies – Hardware and Software					
10.1.7	<i>describes the purpose and effectiveness of operating system tools and utilities to organise and manage the computer.</i> <u><i>Candidates should already have these skills on local, mobile and remote computers</i></u> Structuring data in secondary storage and cloud storage: • Management of desktop • Management of files and folders including sharing and permissions • Archive • Backup • Compress/decompress files System Management: • Installing/uninstalling software (custom and full installation, product keys, activation codes) • Adding devices and device drivers • Scheduling/updating Security features: • Firewall, anti-malware				
10.1.8	<i>states and discusses the implications of the latest computer technologies.</i> Discusses possible impacts of the latest technologies on the different societies by considering gender, race, culture, religion, ethnicity, environmental and economic factors.	11.1.8	<i>states and discusses the implications of the latest computer technologies.</i> Discusses possible impacts of the latest technologies on the different societies by considering gender, race, culture, religion, ethnicity, environmental and economic factors.	12.1.8	<i>states and discusses the implications of the latest computer technologies.</i> Discusses possible impacts of the latest technologies on the different societies by considering gender, race, culture, religion, ethnicity, environmental and economic factors.

Topic 2: Internet and Communication Technologies					
Grade 10		Grade 11		Grade 12	
10.2.1	displays a knowledge of networking in a LAN. Overview of a network: - Describe a network - Network devices/nodes: - Client/workstation - Server - Switch - Router - Firewall - Bounded Media – Types of cabling (UTP and fibre) - Unbounded Media – Microwave, radio wave Classification of networks (PAN, LAN, WAN, GAN): Local area and smaller networks: - General function of an ADSL router – wireless/hotspot, firewall, switch, connection to Internet - Role of a server in a network – any computer that provides a service to clients e.g. authentication, file sharing, email - Role of client in a network – any computer which receives a service (a server and client could be the same machine) - User profiles/rights/permissions to access network resources Reasons for using networks: - Communication - Access to/sharing resources - Centralisation - Data transfer - Productivity Advantages and disadvantages of networks	11.2.1	discusses network configurations, devices and architectures of a LAN and smaller networks. LAN and smaller network concepts: - Bounded connection media (UTP, fibre) - Weaknesses of communication channels (eavesdropping, attenuation, cross talk, EMI) - Unbounded connection media (Bluetooth, wireless, radio waves) - Topology – star, bus, ring, hybrid, mesh - Networking devices such as NIC, switch, router and bridge - Network addressing – compare IPV4 and IPV6 (see 11.4.2), MAC address, DNS, DHCP, ARP - Broadcast vs. point-to-point communication - Format of Packets and Frames for Ethernet and IP addressing including headers (source, destination), payload, frame checks, VLAN tag related to network devices such as switch, bridge and router WLAN - Devices: wireless access point, wireless bridge, wireless router - Wi-Fi and Hotspots - Compare bounded connections (wired) vs wireless connection in terms of bandwidth and speed Extending a LAN - Fibre optic backbone	12.2.1	discusses distributed processing within a LAN. Centralised vs distributed processing: - Thin clients - Fat clients - Smart clients

Topic 2: Internet and Communication Technologies					
		11.2.2	discusses network configurations, devices and architectures of a WAN. WAN: - Gateways, Wi-Fi router - Transmission – satellite radio waves and microwave - Connection technologies - Cellular technologies including latest technologies - Fibre vs ADSL - Difference in usage and bandwidth Protocols: - Email protocols – POP3, SMTP, IMAP - Internet protocols – TCP/IP (UDP vs TCP) - IP frame vs Ethernet frame and associated devices (see 11.2.1) - Web protocols -HTTP, HTTPS - Download protocols – FTP, WebDav	12.2.2	evaluates the effectiveness of cloud computing as an extension to a LAN Sharing concepts: - Peer-to-peer file sharing e.g. BitTorrent - Compare FTP, WebDav and Peer-to-peer file sharing Compare access to remote sites: - Remote access e.g. team viewer - VPN – to provide remote access - Risks and advantages of all methods Internet of Things (IoT): - Concept of devices accessible on the Internet
10.2.3	describes the Internet and connectivity options. Describe the Internet: - Internet as a global network - Basic services on the network (mail, Web sites) Connectivity Options: - Reason for an Internet protocol (IP) address - Hardware to connect – e.g. SIM card (cell), wireless router, personal hotspot, ADSL router	11.2.3	discusses and compares the transmission of data over the Internet. Overview of multimedia as part of Internet technologies: - Download vs. streaming and effect on bandwidth - Video on-demand - VOIP - Podcast/Vodcast Compression technology: - Lossy vs Lossless - Lossy compression of sound files - Compression: Quality vs. bandwidth and speed	12.2.3	discusses the advantages and disadvantages of the deep web. Concepts of: - Deep web and dark web Anonymous Browsing Tools: - Onion router – e.g. Tor - VPN - how it ensures privacy, anonymity

Topic 2: Internet and Communication Technologies					
10.2.4	<i>makes effective use of Internet tools to communicate, collaborate, organise and plan.</i> <u>Candidates should already have these skills on local, mobiles and remote computers</u> Email: • Sends and receives emails • Displays responsible communication styles in terms of netiquette – spam, large file sizes, hoaxes • Monitors emails frequently and organises email account Calendar: • Uses a calendar to schedule and share events Online storage: • Uses online tools to collaborate with notes, terminology lists and tasks (e.g. Google Docs, Office 365 using MS Word) Responsible use of the Internet: • Social Media				
10.2.5	<i>describes the design and functional elements of a Web site.</i> Classification of Web pages/sites: • Web page vs Web site User Interface Design: • Compare usability issues such as readability, navigation (three-click rule), consistency, layout, typography/colour, theme (see 10.4.12) • User interface design as applied to desktop/mobile	11.2.5	<i>describes the evolution of Internet Service Technologies.</i> Overview of the evolution of the Internet in terms of: • Static and dynamic sites: Web 1.0, Web 2.0, Web 3.0, Web 4.0 • Client-side and server-side scripting • Cookies • Web-based applications • Mobile applications – native mobile applications, mobile web applications, hybrid • Plug ins and extensions • Design factors for mobile technology: ▪ Screen size, processing demands, storage demands, bandwidth requirements	12.2.5	<i>critically assesses Internet Service Technologies.</i> Overview of Internet Services technologies: • Location Based Services (LBS) • Global Positioning System (GPS) Discuss and analyse the purpose and effect of cloud computing: • Concepts of cloud computing: • Effect on local hardware needs • Cloud services for application use (G-Suite, Office 365) • Cloud storage for data • Advantages and disadvantages e.g. permissions, security, bandwidth • Cloud licensing vs other licencing models (10.1.6) • Ownership of data Security services: • Public and private key encryption and SSL • Digital certificates • Digital currency: ▪ Block chain technology ▪ Distributed database (decentralisation) ▪ Transaction and ledger

Topic 2: Internet and Communication Technologies					
10.2.6	<i>navigates the Internet in order to retrieve information.</i> Describe the WWW: • Web address/uniform resource locator (URL) Searching Engines: • Defines a search engine with examples • Applies Boolean logic to search criteria (see 10.4.4)	11.2.6	<i>efficiently searches the Internet for information.</i> Search Engines: • Performs searches for text, sound (e.g. Shazam) and images, reverse image lookup • Performs searches using advanced search criteria using logical operators (see 10.4.4, 11.4.4)	12.2.6	<i>describes improving Web site reception for search engines.</i> Improve searching: • Search Engine Optimisation (SEO)
		11.2.7	<i>demonstrates an ability to identify sources of errors and propose solutions.</i> Sources of errors: • Human error – GIGO, input errors • Arithmetic errors – rounding, truncating, fixed number bits, overflow (see 11.4.2) • Data transmission errors – cable problems • Programming errors – undetected logical errors Solutions for errors: • Verification vs validation • Techniques for input - barcode scanner, QR codes, keyboard vs GUI design (drop down boxes, check boxes etc.), RFID, biometric input, Optical Character Recognition (OCR) • Types of checks for data validation – Presence check, range check, uniqueness check, length check, type check, logical check, check digit, check sum (see 11.4.8, 11.4.14) • Data transmission check - parity		

Topic 2: Internet and Communication Technologies					

Topic 3: Social and Ethical issues			
Grade 10		Grade 11	
10.3.1	describes the reasons for and effects of using computers. Economic reasons for using computers: - Saving paper, labour, communication costs, efficiency, accuracy, reliability Digital divide: - Describes the digital divide - Reasons for the digital divide Ergonomics, health issues: - Effects of RSI, eye strain, carpal tunnel Green computing issues: - Green use – minimise electrical consumption - Green disposal – responsible disposal of computers, cartridges, repurposing and recycling	11.3.1	examines the effects of the use of computers and information across a range of application areas. Effect on workplace and employment practices: - Acceptable User Policy (AUP) - Decentralisation of the work place - Replacement of work force - robotics, artificial intelligence, UAV such as drones Computer crime and its effects (see 11.2.8): - Social Engineering - shoulder surfing, dumpster diving, phishing, trojan horse, reverse social engineering or role playing, use of social media for social engineering - Hackers, crackers and virus authors - Computer crimes such as theft of hardware, theft of software, theft of information, identity theft, bandwidth theft, theft of time and services Suggest safeguards against computer crimes, threats and criminals (see 11.2.8) Explains the consequences of inaccurate information on society and provides criteria to evaluate data sources including Web sites - Affiliation (e.g. who supports the source?) - Audience (e.g. level at which it is written/who is it intended for?) - Authority (e.g. who is the author and what are his/her credentials?) - Content (e.g. organisation of content and working links) - Currency (e.g. is the information on the source up-to-date?) - Design (e.g. is it easy to navigate and visually pleasing? How quickly does it download?) - Objectivity (e.g. does it reflect any preconceptions?)
			Grade 12 evaluates the effects of the use of computers and information across a range of application areas Green Computing: - Identifies the negative the environmental impact of the use of computers and provides solutions Explain how computers provide solutions to issues of global importance such as: - Use of artificial intelligence to detect patterns for terrorist activity, earthquakes, tsunami, social media propaganda - Describe the effects of digital communication and technology - Availability of personal information vs misuse of personal information, - Right to access vs. right to privacy - Big data: sources and accumulation, decision making - Digital heritage (legal rights to your data/applications both local and remote upon death) - Social, political (e.g. Fake News), environmental - List and discuss issues regarding privacy and information sharing e.g. Google Drive and Drop Box, movies music - Cyber bullying, illegal distribution of child pornography, harmful or explicit text or multimedia images - Laws – Sexual Offences Act, Protection from Harassment Act, Electronic Communications and Transactions Act, Protection of Personal Information act (POPI) - Discuss the danger of errors in computers controlling equipment e.g. hospitals, motor cars, UAV - Internet of Things (IoT) – implications for security and privacy - Deep and dark web: - Risks and dangers – why it should NOT be used - Possible application in countries where access to social media and news networks are restricted - Implications and effects of cryptocurrencies

Topic 4 – Data and Information Management and Solution Development

Core concepts include data, data structure, algorithms, problem solving and software creation. These are interrelated and build upon each other. Candidates writing the final examinations are expected to have mastered the concepts and skills learnt in previous years.

Topic 4 – Data and Information Management and Solution Development					
Grade 10		Grade 11		Grade 12	
10.4.1	Problem Solving: demonstrates an ability to solve simple problems using computational thinking. Solve simple problems using these four stages (note that they are iterative) Decomposition - Understand the problem - State in own words - Break into smaller parts using IPO and methods Pattern recognition: - Identify similar problems that have been solved previously e.g. input similar types of variables, perform similar calculations and/or similar output - Identify repetitions in the solution (loops and methods) Abstraction: - Identify and ignore the details that are not of interest to this problem - Identify the relevant parts of the problem (some of the decomposed parts are not relevant to the solution) - Represent data using appropriate types and structures - Break solution into parts that once solved can be dealt with as a single abstract concept e.g. IPO – each is a separate part Algorithm: - Combine abstract parts to create a solution - Represent in pseudocode and flowcharts - Code and test the solution Evaluate: - Analyse the solution and determine any errors or inefficiencies including possible improvements - NOTE – Efficiency can refer to: - Code – no unnecessary duplication of code - Execution – no unnecessary processing	11.4.1	Problem Solving: demonstrates an ability to solve more complex problems using computational thinking. Solve problems using these four stages (note that they are iterative) Decomposition: - Understand the problem - Break into smaller parts which can be represented by arrays and objects - Separate the interface and backend - Identify the permanent storage Pattern recognition: - Identify similar data that can be represented by arrays and/or classes - Identify similar behaviour that can be represented by methods in or out of a class. - Identify similar objects. - Reuse or adapt existing objects, methods or classes. - Identify and adapt similar user interfaces. Abstraction: - Decide what is relevant and what is of interest to reduce complexity - Design classes to encapsulate fields and methods - Used information hiding to hide complexity, isolate and protect data Algorithm: - Combine abstract parts to create a solution - Represent in pseudocode and flowchart - Code and apply design and testing principles to ensure each part works individually and together. Evaluate: - Analyse the solution and determine any errors or inefficiencies including possible improvements - Code efficiency – use of methods and parameters to avoid duplication - Execution efficiency – execution terminates when an item is found or an array is sorted	12.4.1	Problem Solving: demonstrates an ability to solve more complex problems using computational thinking. Solve simple problems using these four stages (note that they are iterative) Decomposition: - Understand a large problem and identify goals and sub-goals - working with other stakeholders (such as users, members of development/test teams). - Break the problem down into smaller parts that are solvable using concepts covered previously – meeting the goals and sub-goals in the process. Pattern recognition: - Recognise patterns and select the best data structure including existing classes to use based on data, behaviour and goals Abstraction: - Make use of abstraction tools and structures to reduce complexity of the problem and solution such as inheritance Algorithm: - Combine abstract parts to create a solution - Represent in pseudocode and flowchart - Code and apply design and testing principles to ensure each part works individually and together. - Test the solution works with external, existing or previous solution Evaluate: - Analyse the solution and determine any errors or inefficiencies including possible improvements - Code efficiency – reuse of classes and inheritance - Execution efficiency – no unnecessary processing

Topic 4 – Data and Information Management and Solution Development				
Grade 10		Grade 11		Grade 12
10.4.2	Data: represents data in a fixed number of bits. Overview of number systems: - Decimal, binary, hexadecimal - Conversion between decimal, binary and hexadecimal - Number of possible combinations related to number of bits Overview of digital character representation: - ASCII/UTF-8, Unicode	11.4.2	Data: demonstrates an understanding of representing data in a fixed number of bits. Reason for representing data in binary: - Word size (CPU registers), implications of working with a fixed number of bits. Representing integers: - Formulas to calculate maximum and minimum of signed and unsigned integers - Overflow errors and consequences Representing real numbers: - Mantissa and exponent - Overflow errors and consequences - Accuracy – rounding vs truncation as related to real number representation Possible combinations of a fixed number of bits: - IP Addressing - IPV4 vs IPV6 and how they are represented (see 11.2.1) - MAC address – representation in hex - Number of colours of a pixel and screen resolution Shortening techniques: - Use of hexadecimal digits to reduce number of binary digits	

Topic 4 – Data and Information Management and Solution Development					
Grade 10		Grade 11		Grade 12	
10.4.3	Data and Data structures: understands that simple data comes in different forms and that data typing is important. Simple data: - Text (string), char, integer, floating point (real), Boolean (see 10.4.4) - Arithmetic operators related to each type: (+, -, *, /(real division), mod, div(integer division) - Order of precedence with brackets (see 10.4.4 for relational and logical operators) - Appropriate use of each type to store data - e.g. ID number, telephone number is better stored as a string: - Data type ranges - Constant vs a variable - Naming conventions - Conversion between types: - String and numeric types - String and char - Char and integer - Real and integer - Effects of narrowing and widening conversion - Strings: - Length of string - Concatenation of strings - Changing upper and lowercase in chars and strings - Comparing strings Acquire data from user: (see 10.4.12) Use of methods and objects: (see 10.4.5) Applies to a single table database: (see 10.4.10, 10.4.11)	11.4.3	Data and Data structures: analyses a problem, suggests and codes suitable data structures. Date and/or types provided by programming language: - Use methods provided by these classes to appropriately input, validate, store and output dates and times Strings: - Use methods to isolate characters and strings, compare strings, count characters, insert, replace, append, delete characters (see 11.4.8) Static Arrays: - One dimensional arrays - Parallel arrays - Arrays of objects - Compares parallel arrays to arrays of objects - Sorting, searching calculations and array manipulation (see 11.4.8) Acquires data from UI component (see 11.4.12) and text file (see 11.4.7) Creates objects and classes: (see 11.4.5) Applies to a single table database: (see 11.4.11)	12.4.3	Data and Data structures: analyses a problem, suggests and codes suitable data structures. Dynamic Arrays: - Purpose and function - Compare to static arrays DYNAMIC ARRAYS ARE NOT EXAMINABLE IN PRACTICAL EXAMINATION Combination of different designs – for example: - Array of an object in which one of the fields of the object is an array - Array of inherited objects - Array of an object of which one of the fields of the object is another object - Object where each field is an array of two different types of objects etc. Acquires data from common UI components (see 11.4.12) Creates objects and classes: (see 12.4.5) Applies to a multi-table database: (see 12.4.10, 12.4.11)

Topic 4 – Data and Information Management and Solution Development				
Grade 10		Grade 11		Grade 12
10.4.4	Boolean Logic: identifies the basics of Boolean logic and applies this to Boolean expressions. Identifying a Boolean amongst other data types. Relational Operators: • >, >=, <, <= equality = and inequality != / <> Logical operators: • NOT, AND, OR Order of operations and need for brackets: • Evaluates Boolean expressions with multiple conditions • Uses truth tables with a maximum of three variables to evaluate conditions • Applies to search engines (see 10.2.6), programming language (see 10.4.8) and SQL statements (see 10.4.11)	11.4.4	Boolean Logic: applies logical conditions successfully in a programming language, SQL statements and search engines. • Evaluates complex Boolean expressions in search engines (see 11.2.6), programming language (see 11.4.8) and SQL statements (see 11.4.11) • Codes complex Boolean expressions in search engines (see 11.2.6), programming language (see 11.4.8) and SQL statements (see 11.4.11) • Uses truth tables with maximum of four variables to evaluate complex conditions	

Topic 4 – Data and Information Management and Solution Development			
Grade 10		Grade 11	
Grade 12			
10.4.5	Method and OOP: calls basic functions, works with objects from existing classes and their methods Use of basic mathematical functions and applies them in programs to: - Determine minimum and maximum values, sum, average, exponent (power) functions, calculating absolute values, integer arithmetic (calculating dividend and remainder), calculating square roots, rounding functions (whole and to fixed decimal points), calculating random numbers Instantiates objects of existing classes and use related methods: - Purpose of constructor methods - Calls a typed method/function and void method/procedure method	11.4.5	OOP: designs classes and applies object-oriented principles. Designs and implements classes by creating appropriate fields and methods (see 11.4.3): - Fields and methods, static and non-static (also referred to as class fields and methods) - Protected, private and public fields and methods - Constant fields - Constructors to instantiate an object and assign values to fields (including default constructor/parameterised constructor) - Accessor, mutator and toString methods - Encapsulation and information hiding - Method overloading and dynamic binding - Private helper methods - Parameter passing to send data to a method in an object - Return types - Terminology: instances, instantiation, declare - Concept of an object as the backend independent of the user interface/frontend - Class Diagrams to represent the fields and methods of a class and their accessibility - Understand typed methods/functions and void methods/procedures - Differentiates between typed methods/functions and void methods/procedures methods - Know how to make use of typed methods/functions including in an output statement, condition or assignment
12.4.5	OOP: efficiently designs reusable classes and applies object-oriented principles Extended Objects (see 11.4.3): - Fields of complex types e.g. objects or arrays of other objects - Parameter passing and return of complex types - Null objects or null fields in an object Inheritance (see 11.4.3): - Superclass and subclass - Overriding – polymorphism and dynamic binding - Discusses advantages of inheritance - Compares an object of an object vs inherited objects – i.e. when to instantiate an object as a field as opposed to inherit from an object - Type determination (instance of) - Compares data structures and provides advantages		

Topic 4 – Data and Information Management and Solution Development					
Grade 10		Grade 11		Grade 12	
10.4.6	Data Transfer: transfers data between methods. Parameters: - To send data to a method – match number of parameters, type and order in the parameter list Return type: - To return data from a method	11.4.6	Data Transfer: effectively and efficiently transfers data between objects. Apply parameter passing and return values as communication between: - The user interface/frontend (application) and the object class/backend - Methods including private helper methods Use of parameters to send data to methods: - Single variable (primitive type) - Object Typed methods/functions to return data from methods: - Single variable (primitive type) - Object - String with fields separated by a # or similar character - Null Parameters as a mechanism for abstraction – reduce code and make methods more generic (code efficiency) Concept of scope and lifetime of variables, fields and parameters	12.4.6	Data Transfer: effectively and efficiently transfers complex data between objects. Use of parameters to send data to methods: - Array and array of objects Typed methods/functions to return data from methods: - Array and array of objects
		11.4.7	Persistence: uses secondary storage to store, edit and input data. Text files: - Create, append, read - Multiple lines with fields related to the same data structure - Multiple lines with fields related to different data structures - Input into a complex data structure such as an array of objects - Testing if the file exists – exception handling (see 11.4.14)	12.4.7	Persistence: compares secondary storage to store, edit and input data. Concept of JSON files: - Purpose and function of these files - Structure of JSON files for storing and transferring complex data - Compare JSON, text files and databases – advantages and disadvantages 12.4.7 IS NOT EXAMINABLE IN THE PRACTICAL EXAM

Topic 4 – Data and Information Management and Solution Development					
Grade 10		Grade 11		Grade 12	
10.4.8	Algorithm: plans and implements selection and simple looping for a variety of simple algorithms. • Concept of algorithm: • Use pseudocode or flowchart to represent an algorithm (see 10.4.2) • Implements programmatically selection and looping using: if, if ... else, switch/case, for, while, do ... while, repeat until, • Solves general computing problems such as finding smallest, biggest, sum, average, factors and multiples, swapping values, isolating digits of an integer. • Determines best use of different selection structures (if vs switch) • Determines when to use a counting vs. condition loop (i.e. when to use for, when to use while). • Appreciates the difference between a pre-check (while) and post-check (do...while) loop. • Codes nested/cascading selection statements • Codes nested loops with independent internal variable. • Uses methods to abstract the complexity of nested selection structures and nested loops (see code efficiency)	11.4.8	Algorithm: plans and implements solutions in a programming environment for a variety of algorithms. Array manipulation: • Search (sequential and binary) • Sort (selection, improved selection, bubble and bubble sort with a flag) • Insert an element and delete an element • Remove duplicates for arrays of simple data types and objects String manipulation: • Counting words in a string • Isolating words in a string • Removing vowels in string • Encoding/encrypting a string (see 11.2.8) • Changing a full name "Fred John Smith" to "FJ Smith" • Validation of input data such as an ID number (see 11.2.7) • Calculating check digits (see 11.2.7, 11.4.14) Identifies patterns in duplicated code to identify reusable methods and parameters (pattern recognition and code efficiency) (see 11.4.1 and 11.4.6)	12.4.8	Algorithm: plans and implements solutions to simple problems requiring collections of data in a programming environment. • Search, sort, insert, delete, etc. with arrays of objects/extended objects (see 12.4.3, 12.4.5 and 12.4.6).

Topic 4 – Data and Information Management and Solution Development				
Grade 10		Grade 11		Grade 12
		11.4.9	Database: describes the purpose and features of a database and DBMS. Explain the functions of database management system (DBMS): • Data integrity management – accuracy, correctness, currency, completeness, relevance • Security - multiuser access control, encryption, design flaws and programming bugs, SQL injection, Malware infections • Multiuser access control • Backup recovery and management	12.4.9 Stored Data: analyses strategies for storing, protecting and retrieving stored data. Identify threats to quality data: • Corrupted data • Outdated data • Invalidated data • Vulnerability (SQL injection, malware) Understand the storage of data – Data warehousing: • Describe data warehousing • Purpose and uses Understand the extraction of data – Data mining: • Description and purpose • Big data – sources (e.g. social media, activity generated data, logs and audit trails, location-based data) Explain the concept of NoSQL: • Description and example e.g. MongoDB • Compare to SQL
10.4.10	Relational Database: understands and works with a single table relational database. Describes concepts of a relational database: • Field, record, table • Primary keys • Database Schema/structure vs data			12.4.10 Database Design: efficiently designs a multi-table database. Explain the reasons for normalisation: • Data redundancy and repeating groups • Anomalies – Update, Insert and Edit Describe key fields: • Primary keys • Foreign keys • Composite keys Identify database concepts: • Duplicate data, derived data and redundant data • Atomic fields and non-atomic fields Design and create a multi-table relational database: • Normalisation • 1NF, 2NF and 3NF • Data dependencies – partial and transitive • One-to-one, one-to-many and many-to-many relationships • Importance of referential integrity 12.4.10 IS ONLY EXAMINABLE IN SBA AND PAT

Topic 4 – Data and Information Management and Solution Development					
Grade 10		Grade 11		Grade 12	
10.4.11	Database and SQL: create tables and simple queries using a single-table database. Create a single-table database using an application package: - Field types (suitable), sizes, default values - Autonumber primary key - Not null and indexed fields Query using SQL to access a single table database: - SELECT, FROM, WHERE, DISTINCT - LIMIT/TOP - ORDER BY ASC/DESC - Logical operators: NOT, AND, OR, IN (see 10.4.4) - Special operators: BETWEEN, LIKE, IS NULL - Applies Boolean logic to search criteria (10.2.6) Alter a single table in a database using SQL: - INSERT, UPDATE, DELETE with WHERE using logical and special operators	11.4.11	Database and SQL: query a single-table database using more complex SQL statements. Access a single table database using SQL: - GROUP BY - HAVING - Refer to 11.4.4 for multiple conditions - Creating calculated fields: - Concatenating fields - Renaming fields using AS - Generating random numbers - Formatting with ROUND, INT, FLOOR, CEILING - Casting a field (conversion between types) - Mathematical operators including MOD and DIV/integer division - Aggregate functions: SUM, AVERAGE, MIN, MAX, COUNT - Common date functions NOW, YEAR, MONTH, TIME, DATE, HOUR, MINUTE and DAY - Accurate DATE calculations for age - String functions: Finding length and extracting part of a string e.g. LENGTH MID, LEFT, RIGHT, SUBSTR (or equivalent to find the position of a character in a string) - Combination use of the above Alter a single table in a database using SQL: - INSERT with limited fields or Autonumber primary key - SQL statements with Boolean conditions mentioned in 11.4.4	12.4.11	Database and SQL: queries a multi-table database using complex SQL statements. Create simple subqueries (nested queries) Use SQL to access a multi-table database using primary and foreign keys: - INNER JOIN or WHERE to combine multiple tables where keys match - LEFT or RIGHT JOIN to include extra records from one table - LEFT or RIGHT JOIN or NOT IN to find records not related to another table Alter a single table in a database using SQL: - INSERT with a SELECT where the SELECT statement can be any combination of SQL statements Access a multi-table database through programming language constructs – ONLY EXAMINABLE IN THE PAT - Setup a connection or connect to a database using a database connection class by providing a path in code statements - Query and edit a multi-table database using appropriate SQL constructs - Access and modify fields and records - Represents in primary memory using appropriate data structures

Topic 4 – Data and Information Management and Solution Development				
Grade 10		Grade 11		Grade 12
10.4.12	Software Creation: User Interface: Design and codes a simple user interface: - Creates a simple text-based user interface - Codes appropriate user prompts and error messages based on simple data validation Describe the principles of good user interface: design from the visual perspective (see 10.2.5): - Structure – UI should be organised meaningfully - Simplicity – UI not cluttered, tasks easy to achieve - Visibility – All options available for user but no extra distracting information - Feedback – keep users informed of actions, states, errors relevant and of interest to the user - Tolerance – design should be flexible and tolerant. A user will make mistakes, however, the design should prevent incorrect input - Reuse – reuse behaviours and components to maintain consistency, reducing time for users to rethink and remember Compares Desktop UI to a Mobile Application UI	11.4.12	Software Creation: implements good design in user interfaces. Applies good user interface design principles described in 10.4.12 including: - Descriptive error messages in programming (the error message returned should indicate a solution) - Appropriate user prompts and error messages based on exceptions caught or situation required - Appropriate use of metaphors or images (e.g. a picture of a printer on a print button) - Apply consistent behaviour, which makes use of long-term memory e.g. always using F1 for Help or ESC to stop a process. There are certain functions that have become de facto standards. Clear and helpful error messages. - Create uncluttered screens with effective use of colour - Use components effectively to provide data validation such as drop-down boxes, calendar. 11.4.12 – NOT EXAMINABLE IN THE PRACTICAL EXAM	
10.4.13	Software Creation: suggests ways in which well-known software can be methodically tested for robustness. - Select appropriate standard, extreme and abnormal data to test a program - Use trace tables to test program logic and identify errors - Understand the difference between and causes of syntax, runtime, logical errors	11.4.13	Software Creation: masters basic techniques for debugging and testing of programs. - Use the debugger facilities of a programming language including watches, traces and breakpoints. - Use trace tables to test program logic, identify errors and evaluate program execution efficiency. - Identify and corrects syntax, runtime and logical errors - Understand the value of generated test data - Test program code using standard, extreme and abnormal data	

Topic 4 – Data and Information Management and Solution Development				
Grade 10		Grade 11		Grade 12
		11.4.14	Software Creation: understands the process of data validation and can describe and code data validation techniques - Understand the reasons for data validation e.g. prevent entry of erroneous data - Use exception handling to deal with errors - Code validation checks: presence check, range check, uniqueness check, length check, type check, logical check, check digit, check sum using conditional loops where appropriate. (see 11.2.7)	
		11.4.15	Software Creation: identifies the correct use of appropriate help in any application. Describe when and where help should be available in an application e.g. mobile applications do not need user manuals 11.4.15 – NOT EXAMINABLE IN THE PRACTICAL EXAM	Software Creation: implements an effective online help system for an existing or new software application. - Include user help in a programming environment, e.g. context sensitive help, menus, FAQs - Include APIs and comments in programming project such as APIs. 12.4.15 – ONLY EXAMINABLE IN THE PAT
				Software Creation: produces well-written and well-presented documentation for an existing or new software application. Produce a Specification Document, Design Document, Technical Document and Testing Document for a project. 12.4.16 – ONLY EXAMINABLE IN THE PAT

APPENDIX H



**NATIONAL SENIOR CERTIFICATE EXAMINATION
INFORMATION TECHNOLOGY
LETTER FROM THE HIGH SCHOOL HEAD AND IT TEACHER/S**

Name of School	
Centre Number	
Completed by	
Designation	

The IEB
P O Box 875
Highlands North
2037

Dear IEB

RE: PRACTICAL EXAMINATION DECLARATION FOR INFORMATION TECHNOLOGY

We certify that we have ensured that	Cross your response	
All the work in each candidate's folder has been copied from the examination location to the IEB CDs without any modification whatsoever.	YES	NO
All the work in each candidate's folder is the work of that learner only.	YES	NO
One hard copy of each candidate's practical examination, labelled with the examination number sticker has been sent to the IEB.	YES	NO
No candidate had access to the Internet during the examination.	YES	NO
No help or assistance at all was given to or allowed to be given to any candidate other than that of a technical nature as stated in the Examination Instructions.	YES	NO
A copy of each candidate's work has been saved on the school system and will be kept at the school on relevant storage media until 28 February of the following year when the remarking process has been completed.	YES	NO

IT EDUCATOR/S

HEAD

DATE

DATE

APPENDIX I



**NATIONAL SENIOR CERTIFICATE EXAMINATION
INFORMATION TECHNOLOGY
APPLICATION FOR ALTERNATE LANGUAGE FOR PAT**

Please fill in each of the following fields:

1. Centre Number

2. School Name

3. Proposed Language

4. Candidates name and Examination number

NAME	EXAMINATION NUMBER

5. Name of educator responsible for teaching proposed language

6. Qualifications of educator

7. Years of experience teaching the proposed language

8. Person responsible for internal moderation

9. Qualifications of internal moderator

10. Years of teaching experience of internal moderator.

11. Motivation for use of proposed language.

12. Agree to send PAT for National moderation if there is no Regional Moderator.

YES	NO
-----	----

Educator Signature

Principal Signature

PROCEDURE

1. Email this document to the IT Subject Specialist.
2. This will be sent to the SBA & PAT moderator.
3. Approval will be either granted or not, based on the information and motivation provided in this document.
4. The decision will be communicated to the applicant within two weeks of the application being received.
5. Send the PATs of the above candidates to National Moderation.